APPLICATION FOR UNITED STATES LETTERS PATENT

INVENTORS: Sang Ho LEE and Jang Keun OH

TITLE: METHOD FOR MEASURING QUANTITY OF USAGE OF CPU

ATTORNEYS: FLESHNER & KIM, LLP
&   P. O. Box 221200
ADDRESS: Chantilly, VA 20153-1200

DOCKET NO.:   P-212

METHOD FOR MEASURING QUANTITY OF USAGE OF CPU

BACKGROUND OF THE INVENTION

1. Field of the Invention

[1] The present invention relates to a method for measuring a quantity of usage of a device, in particular, to a method for measuring a quantity of usage of a CPU.

2. Background of the Related Art

[2] There are various kinds of algorithms for predicting a quantity of usage of a system in the related art. In general, more complicated algorithms yield more credible predictions; less complicated algorithms yield less credible predictions.

[3] In addition, a method for calculating a quantity of usage of a CPU can be differentiated in accordance with an operating system. For example, in a MS-Windows 9x system, system usage of a CPU is updated by the CPU itself using its own algorithm. However, it is recommended to use the registry information when an application program is developed.

[4] Furthermore, because an Operating System (OS) kernel takes charge of the CPU multi-tasking in all threads in a present ready-to-run state, the OS kernel generally knows whether there are OS threads in the ready-to-run state or an action state (all threads do not wait for certain event to resume execution). For example, when the OS determines there are no

threads in the ready-to-run state, a value of a power management timer is read, and the CPU is maintained in sleep mode. In this mode the power management timer operates independently from the CPU, and measures accurately time duration of reads (regular increase with a fixed rate of 3.579545Mhz). After that, the OS reads the value of the power management timer when the CPU is waken up.

[5] In addition, measuring time difference between initial timer read in the sleep mode of the CPU and second timer read in the wake mode is a measure of idle time of the CPU. In other words, the CPU idle time can be measured by dividing a total of the difference between the second timer and initial timer for a large sampling interval, by the sampling interval. It can be described as in Equations 1 and 2,

[Equation 1]

Idle Ticks = Sum (across sampling interval)[Second Timer Read-Initial Timer Read]

[Equation 2]

CPU Idle(%) = Idle Ticks x Tick Period(s)/Sampling Interval(s) x 100%

Herein, the first timer read describes a processor in sleep mode (initial timer read), and the second timer read describes the CPU is in the wake mode.

[6] In contrast, related art algorithms yield large values of CPU usage, even

3

though the system does not perform an operation. In addition, in the related art, there is no time interval information available for updating a quantity of usage of a CPU, accordingly it is not appropriate for measuring a quantity of usage of a CPU in short time. In addition, in the related art, because an algorithm has to be amended in order to adapt it to the MS-Windows system or a complicated code is required, it degrades the MS-Windows system performance.

SUMMARY OF THE INVENTION

[7] An object of the invention is to solve at least the above problems and/or disadvantages and to provide at least the advantages described hereinafter.

[8] An object of the present invention is to provide a method for measuring a quantity of usage of a CPU which is capable of getting directly a result by using various functions provided by the operating system on the behalf of a registry storing a quantity of usage of a CPU inside the system.

[9] In order to achieve the object of the present invention, the method for measuring a quantity of usage of the CPU comprises reading execution time of all threads excluding a system thread with a known time interval, adding the values, subtracting a total of the execution time of the former stored thread from the grand total, and measuring a quantity of usage of a CPU by dividing the subtracted execution time by the known time interval.

[10]   To achieve at least the above objects and advantages in a whole or in parts and in accordance with the purpose of the present invention, as embodied and broadly

4

described, a method for measuring a quantity of usage of a CPU in a system, including reading execution time of all threads excluding a system thread with a certain timer time interval, adding the read values, subtracting a total of execution time of the former stored thread from the grand total, and measuring a quantity of usage of a CPU by dividing the subtracted execution time of the thread by the certain timer time.

[11]   To further achieve at least the above objects and advantages in a whole or in parts and in accordance with the purpose of the present invention, as embodied and broadly described, a method for measuring CPU usage, including reading an execution time of a thread over a time interval, adding the execution times to obtain a grand total, reading a total execution time for a previously stored thread, subtracting the total execution time for the previously stored thread from the grand total to obtain a result, and outputting the result.

[12]   Additional advantages, objects, and features of the invention will be set forth in part in the description which follows and in part will become apparent to those having ordinary skill in the art upon examination of the following or may be learned from practice of the invention. The objects and advantages of the invention may be realized and attained as particularly pointed out in the appended claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[13]   The invention will be described in detail with reference to the following drawings in which like reference numerals refer to like elements wherein:

[14]   FIG.1 is a detailed flow chart illustrating a method for measuring a quantity of usage of a CPU in accordance with an embodiment of the present invention.

[15]   FIG.2 illustrates an interface between a procedure for calculating a quantity of usage of a CPU and an outer program.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[16]   Hereinafter, a thread and a handle in accordance with the present invention will be described, and an embodiment in accordance with the present invention will now be described in detail with reference to accompanying FIGS. 1 and 2. First, a thread used in a Windows system is an execution unit in an application program. The each thread is combined with CPU commands, and a set of CPU registers, stacks, etc.

[17]   In the Windows OS, a process can have several threads, and the thread can make other threads again. In the meantime, a kernel as the core of the Windows OS uses a service called a scheduler allocating CPU time to each execution thread. When a process is getting made, the system makes automatically one thread for the process. The thread is called a 'basic thread' of the process, the basic thread can make additional threads, and the each additional thread can make threads also.

[18]   In addition, in the Windows system, it is not possible to approach the kernel, a Graphic Device Interface (GDI) or other object directly, but it is possible through a handle. In general, because the objects have mobility in a memory by a memory manager of the

Windows, it is possible to search and approach the objects only through their handles.

[19] A first embodiment in accordance with the present invention will now be described in detail with reference to accompanying FIGS.1 and 2. FIG.1 is a detailed flow chart illustrating a method for measuring a quantity of usage of a CPU in accordance with a preferred embodiment of the present invention. In other words, a quantity of usage of a CPU is calculated by using a system service provided from MS-Windows. In Windows, a VMM (Virtual Machine Manager) service usable in an embodiment of a VDD (Virtual Device Driver) is provided. In other words, a service related to a timer and threads is used in the VMM service, and preferably operating at a ring 0 level.

[20] In general, a program operated in the system is classified as a thread, and the thread is a minimum unit of execution. Herein, the execution means the CPU is used. As described in FIG.1, a variable (SUM) storing the sum of the execution time of all threads excluding the system thread is initialized as in step S10. A handle value Hd1 is then read as a pointer of the system thread in step S11, and the next thread handle value Hd2 is read in step S12. After reading the handle value Hd1 of the system thread and handle value Hd2 of the next thread, they are compared in step S13 to determine whether the read values are the same. In other words, it is judged whether the all thread values have been read by comparing the handle value read most recently with the handle value of the system thread.

[21] Herein, the thread handles are executed repeatedly by scheduling of the operating system. The operating system also manages information about the thread and

7

execution of it. In addition, Windows has the VMM service information about the thread execution. For example, the VMM service comprises a function (Get_Sys_Thread_Handle) finding a handle of the first thread (system thread), a function (Get_Next_Thread_Handle) finding a handle of the next thread, and a function (_GetThreadExecTime) finding execution time after a thread generation etc.

[22]   After that, in the comparing process of step S13 of the thread handle, when the thread handle value of the system is different from the next thread handle value, execution time of the next thread handle up to the present is read in step S14. The execution time is then added to the variable SUM in step S15. In judging whether the all thread handles are read by comparing the read values, when the thread handle value of the system is different from the next thread handle value, the above-mentioned process is performed repeatedly until the next thread value approaches to the thread handle value of the system.

[23]   When the thread handle value of the system is the same as the last thread handle value, the execution time of the all thread handles excluding the handle value of the system thread is stored in the variable SUM. Accordingly, the sum execution time (before SUM) of the thread handles stored formerly is subtracted from the total execution time (present SUM) of the thread handles stored in the variable SUM, and the subtracted value is divided by the time interval at step S16. Herein, the value divided by the time interval is a quantity of usage of a CPU (CPU_USAGE).

[24]   For example, when the total execution time of the formerly stored

8

thread handles is 100 seconds and the total execution time of the present thread handles is 105 seconds after a 10 second interval, a value found by dividing 10 seconds by 5 seconds as the ratio between the execution time (100 seconds) of the former stored thread handle and the execution time (105 seconds) of the present execution thread handle is 50% as the quantity of usage of the CPU (CPU_USAGE). In other words, the execution time for the interval is found by calculating a total of the execution time of all the thread handles excluding the handle of the system thread at each interval by using the VMM service, and subtracting the former calculated total execution time from the grand total. And, a quantity of usage of a system is measured by dividing the found execution time for the interval by the interval.

[25]   In the meantime, when the interval is set shorter, the calculated quantity of usage of the CPU shows more sudden change than a case when the interval is set longer, in order to decrease the sudden variation, a quantity of usage of a CPU is compensated by finding an average value between the former calculated quantities of usage of a CPU (CPU_USAGE_PREV) and the present calculated quantity of usage of a CPU (CPU_USAGE_NOW) in step S18. In other words, when the interval is short in step S17, the calculated quantity of usage of the CPU shows a sudden variation. It shows sudden variation also however, in an application construction responding sensitively in accordance with a quantity of usage of a CPU. Accordingly, the above-described sudden variation of the calculated quantity of usage of the CPU can be prevented by compensating the calculated quantity of usage of the CPU. On the contrary, when the interval is sufficiently long, the

9

calculated quantity of usage of the CPU is maintained as it is in step S19. The total procedure ends after storing the usage values and reserving the time for the next procedure.

[26]   FIG.2 illustrates an interface between a procedure for calculating a quantity of usage of a CPU and another program. It is possible to update a quantity of usage of a CPU calculated by the algorithm of FIG.1 to a registry or to access in the Windows program through an interface between an application program (win App) and a VxD or other device driver. As described above, the calculated quantity of usage of the CPU of FIG.1 is updated continually to a memory. In the other device driver (VxD), a quantity of usage of a CPU can be gotten through an interfacing method between device drivers. In addition, a method reading a quantity of usage of a CPU directly by using the application program and device I/O control can be used.

[27]   As described above, the method for measuring the quantity of usage of the CPU in accordance with the present invention can measure a quantity of usage of a CPU with higher confidence by using various functions provided by an operating system. In addition, a power consumption in a system (computer) decreases by adjusting a clock pulse of a CPU with the method for measuring the quantity of usage of the CPU in accordance with the present invention. In addition, the method for measuring the quantity of usage of the CPU in accordance with the present invention can be adapted to various applications based on the system execution requirement, and it is very useful for an application monitoring and reporting a load of a CPU in accordance with an operation state of a system.

[28]   In addition, because the method for measuring a quantity of usage of a CPU in accordance with the present invention is embodied in a device driver (VxD) level, control of a system is useful. In addition, because the method for measuring a quantity of usage of a CPU in accordance with the present invention uses a basic service provided from MS-Windows, there is no need to amend an algorithm in order to adapt it for the other MS-Windows nor does it require a complicated code, and the present invention can measure a quantity of usage of a CPU simply, and without lowering performance of the MS-Windows system.

[29]   The foregoing embodiments and advantages are merely exemplary and are not to be construed as limiting the present invention. The present teaching can be readily applied to other types of apparatuses. The description of the present invention is intended to be illustrative, and not to limit the scope of the claims. Many alternatives, modifications, and variations will be apparent to those skilled in the art.